Architecture and Design for a Space Based Internet Emulation System

Gary J. Minden, Joseph B. Evans, Leon S. Searl and Pooja J. Wagh

{gminden, evans, searl, pjwagh}@ittc.ku.edu

Information and Telecommunication Technology Center

Department of Electrical Engineering and Computer Science

The University of Kansas

Lawrence, KS 60045

Abstract- We describe the architecture and design for a Space Based Internet (SBI) emulation system. SBI envisions that each Earth Observing System satellite participates in a Space Based Internet. That is, that each satellite is capable of originating traffic, terminating traffic, and most importantly, is capable of switching traffic traveling between other satellites and the ground. To achieve a SBI, each observation satellite carries a communications system with several channels or beams (RF or optical) and a network layer switch to route traffic between the several channels and local payload. We describe the motivation for an emulation system, our architecture, and design.

1 INTRODUCTION

Earth Observing System (EOS) [1]satellites have enhanced our understanding of the Earth and surrounding space through remote sensing and communication of those results back to the Earth. EOS supports a coordinated series of satellites for long-term global observations.

Earth Observation Satellites use a number of technologies to collect observations, store raw data and communicate the information to earth stations. The techniques include direct transmission, storage and deferred transmission and relay through communication satellites. EOS satellites launched by NASA [2] transmit their information to TDRSS [3] (*Tracking and Data Relay Satellite System)* that relay this information to Earth stations. TDRSS satellites are located in GEO orbits and therefore have a wider coverage area of the Earth than the EOS satellites. Currently, there are 6 TDRSS satellites, which provide complete global coverage.

Each EOS satellite has a fixed time slot to transmit its information to TDRSS. So the EOS satellites have to store information onboard until they can transmit data to TDRSS. This requires high data rate and high capacity recorders on the satellites to store observational data. Also, the communications systems on the satellites are *satellite specific*, which means that each satellite should have its own communication frequency, protocol and command structure. This approach leads to incompatible and non-reusable communications components.

Our work for the SBI project aims at establishing an internet between the satellites and ground stations. Each satellite in the SBI system will be able to route data and switch traffic between other satellites and ground stations.

The advantages of a Space Based Internet in which each observation satellite takes part are several folds:

1. The system is scalable; as each satellite is launched it adds capacity to the SBI

2. Standard SBI software modules can be designed, constructed and deployed to minimize individual satellite cost,

3. SBI modules can be enhanced over time as technology advances,

4. Future missions are likely to be "joint" across multiple platforms and programs calling for an integrated communications capability, and

5.Communications between satellites will be necessary for coordinated data collection, especially for unique events.

Our approach is to design initial network architecture for a Space Based Internet and to evaluate the architecture using emulation techniques. The emulation system models a satellite system by representing each satellite or ground station with a separate computer (PC-based); emulating communications with Ethernet and delay; and controlling the test scenario through management software. By emulating each satellite with a single PC, it is easy to (a) generate traffic based on the type and utilization of the instrument(s), (b) execute actual applications, and (c) test a multiple routing and control approaches. The remaining part of this paper outlines the architecture and the design of our SBI Emulation System.

We first describe the background and motivation for designing this system. We then describe the structure of a satellite node, the operations center, and the emulation manager. Finally, we provide a short description of our emulation system.

2 BACKGROUND WORK

Our approach to understanding and designing SBI Emulation System is based on our work on Rapidly Deployable Radio Network (RDRN) [5]. The RDRN project, sponsored by DARPA, developed network control programs and communication systems to demonstrate a mobile wireless network.

The RDRN system utilized wireless ATM technology to provide an adaptive and re-configurable network [5]. It interoperated with IP based networks via Ethernet or fiber optic cable and provided multi-hop operation over wireless nodes as far as 10 Km apart. Each RDRN node supported several RF communications channels. A network control program monitors the location of nodes, establishes the network topology, and determined the routes. The nodes determined their geographic location from Global Positioning System receivers.

The RDRN emulation system was designed to test the RDRN network control for multiple, mobile scenarios. Emulation enabled testing many more scenarios; avoided the cost of building and deploying a large, mobile, wireless network; allowed repeated experiments to check bug fixes and compute statistical confidence intervals; and allowed the use of actual application traffic in addition to artificial traffic generators. Evaluation and testing of RDRN systems much larger than those practicable through actual deployment was possible.

The emulation system consisted of RDRN nodes, represented by one PC for each node, and an Emulation Manager. The nodes were linked by Ethernet connections through a managed Ethernet switch. The Emulation Manager performed the following functions: (a) determined the location of each node from a pre-established scenario; (b) transmitted each node's position in GPS messages to each node via a management Ethernet; (c) controlled the communication data network (Ethernet) between the nodes; (d) emulated a low bandwidth order wire system the nodes used to exchange information between themselves. The nodes were responsible for executing the RDRN network control program. They received their own GPS location, exchange their location and the location of other nodes via the order wire system, computed a network topology and routing tables, and established communication channels via the managed Ethernet. The Emulation Manager could block order wire transmission and/or setting up communication channels if it determined terrain or distance blocked the communications.

3 SBI EMULATION SYSTEM

3.1 SBI Emulation Architecture

The SBI Emulation System design extends the land-based RDRN Emulation System to a space-based system. SBI systems contain (a) EOS satellites, (b) a data relay satellites, and (c) a ground stations.

EOS satellites collect data and transmit data via one or more communications channels. Currently, we assume satellites have a few (1-4) communications channels, either RF or optical, and general bandwidth up to 100 Mbps with a few 400 Mbps links. Initially, we do not account for antenna/link constraints. For SBI satellites that are under consideration, the instrument rates range from a few bits per second to at least 150Mbps. These satellites are mostly LEO satellites, which orbit at an altitude less than 2000 Km above the earth's surface.

Data relay satellites are used solely for relay purposes. These satellites have high data rates, multiple communications channels, and act as router satellites switching traffic between other satellites and ground stations. A relay satellite is likely to be in either a GEO (36,000 Km) or MEO (10,000 Km) orbit.

Ground stations are used to record collected data, transmit data to research groups, and manage and control the satellites and data collection process. The data collection process will be driven by scheduled collection periods, investigator requests, and 'collection opportunities,' such as a volcano eruption or strong storm system. Since collection requests are generally known ahead of time and satellite positions can readily be calculated ahead of time, ground stations will be able to pre-compute routing tables and traffic loads. Multiple ground stations are supported.

Traffic between EOS satellites, data relay satellites, and ground stations is carried by the Internet Protocol (IP).

3.2 SBI Emulation Design

Physical PC computers running the Linux OS represent the satellites and ground stations in a SBI. In addition to the emulation of physical system components, there is at least one Emulation Manager node that controls the experiment and sets the scenario.

SBI Emulation System software consists of three parts: (a) SBI Node Software, (b) SBI Emulation Software, and (c) SBI Operations Software.

SBI Node Software refers to the software modules that would execute on the satellites and the ground stations in an actual satellite system. This software is resident on the SBI nodes representing EOS satellites, data relay satellites, ground stations, and satellite operational nodes. The operational node software contains software modules that implement adaptive algorithms for configuring the entire network topology. It computes the optimum connections between the nodes based on the scenario obtained from the emulation manager, computes the routing tables for SBI nodes, and communicates the configuration information to the other SBI nodes.

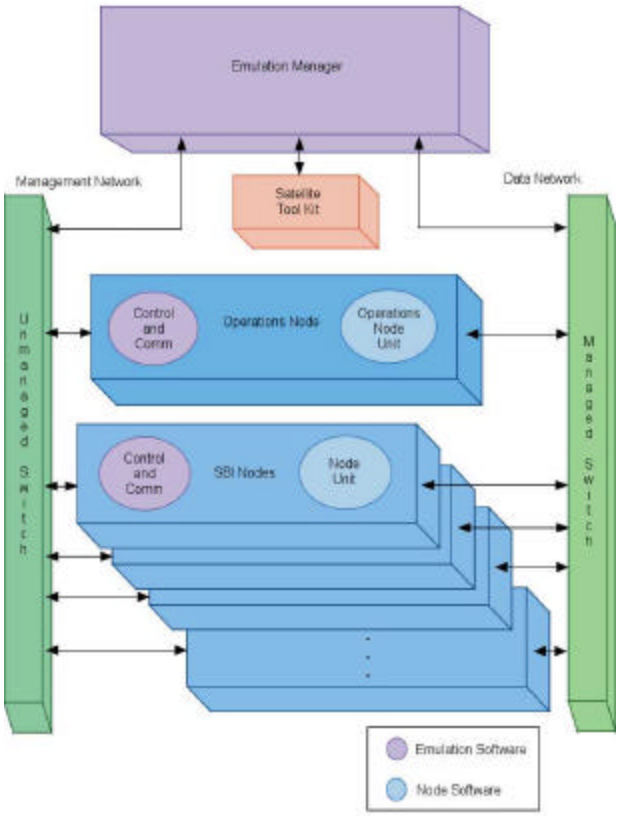

Fig 1. Space Based Internet Emulation System Architecture.

SBI Emulation Software (a) emulates those portions of the SBI system that cannot be easily constructed or deployed, (b) the communications channels between the satellites and ground stations, and (c) traffic generators and controllers to emulate data collection.

Ideally, the SBI Node Software and the SBI Emulation Software would be separate components. Then, one could easily replace the SBI Emulation Software with operational software, and launch. However, these software systems are linked in the emulation system.

On a typical SBI satellite we expect the following software components are present: (a) data collection instruments, (b) Internet Protocol communication systems, and (c) multi-channel communications systems. A data relay satellite or ground station would implement only the (b) and (c) components. In addition, a SBI system would have one or more operation centers to manage the satellites. Operation centers are likely to be co-located with ground stations.

3.2.1 SBI Node Emulation Software

We will describe the SBI software components from the application level down to the communication level. Some components will be identical or very similar to software components that would be present on an actual satellite. Other components will be emulations of actual satellite components. Figure 2 illustrates the SBI Satellite Node Software architecture.

A SBI satellite is under command and control of an Operations Center. The Operations Center performs two major functions: (a) scheduling instrument operation and (b) computing SBI routing structures. In the first stage of this project we implement central control of routing, in later stages we will consider distributed routing architectures.

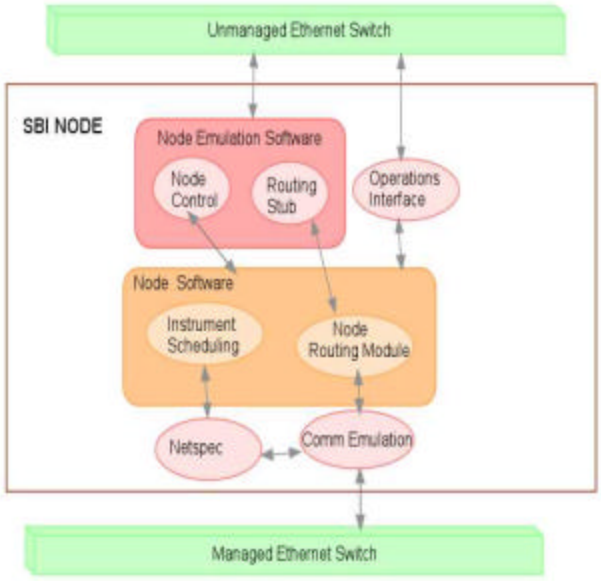

Fig 2. SBI Node Software Modules

On the satellite, the *Instrument Scheduling Component* receives instrument-scheduling commands on the Operations Interface. These commands initiate data collection, terminate data collection, or set instrument parameters. We use *NetSpec* [4] to generate and measure the instrument data stream. NetSpec is a general purpose network traffic source and sink tool that supports many traffic models and collects and reports network statistics in an organized manner. We have compiled traffic characteristics for many satellite instrument packages.

The *Node Routing Component* also receives commands from the operations center via the Operations Interface. This component updates the local satellite routing tables. Each satellite is capable of switching traffic from the local satellite to other satellites (and eventually to a ground station) and carrying transit traffic. The routing component is implemented on the emulation system using standard capabilities of the Linux OS. All routing is Internet Protocol (IP) based.

The *Communications Component* is the key component that emulates the communications link between satellites or the ground station. Traffic generated locally or transit traffic is sent through this component to nearby satellites/ground stations based on the routing tables. We emulate the communications link between satellites with a switched Ethernet and multiple Ethernet drivers within the operating system. We insert a Virtual Ethernet driver between the

Internet Protocol layer and the physical Ethernet driver. The Virtual Ethernet (VETH) layer enables us to establish multiple point-to-point links between satellite emulation nodes. The multiple links are possible because we have obtained a unique Ethernet identifier and use the least significant bytes to identify a specific channel. Thus, each VETH in the system has unique MAC address. By 'labeling' each point-to-point packet, we can control connections between nodes with a managed Ethernet switch. The Emulation Manager (described below) controls the Ethernet switch to enable or disable connections between nodes.

The Virtual Ethernet (VETH) layer implements two important communications link characteristics. First, VETH implements a constant bit rate service by 'metering' packets from the emulated satellite system onto the Ethernet. When a link is setup based on a scenario, the link bandwidth is specified and established via the Operations Interface.

Second, the VETH implements a propagation delay based on end-point satellite position. Propagation delays (one-way) vary from a few milli-seconds to 250 milli-seconds. (We only consider satellites 'inside' geo-stationary orbits.) By emulating propagation delay on each point-to-point line, our system is scalable in the number of satellites. The propagation delay is variable during the emulation. This accounts for two satellites approaching or receding from each other.

3.2.2 Operations Node Software

The Central Operations Node is responsible for determining the network topology, configuring the routing tables for all the nodes, and determining when satellite instruments collect data. Figure 3 shows the different modules of the Operations Node Software.

The Topology Module determines the connections for each node and the *Routing Module* computes the routing tables for each node and sends those tables to the satellites. The *Instrument Scheduling Module* is responsible for scheduling the satellite instruments on all the nodes. Routes will account for the traffic load, satellite location, and ground-station location. We expect routes will be computed every few minutes. The routing and scheduling information is transmitted to the respective nodes via the Operations Interface.

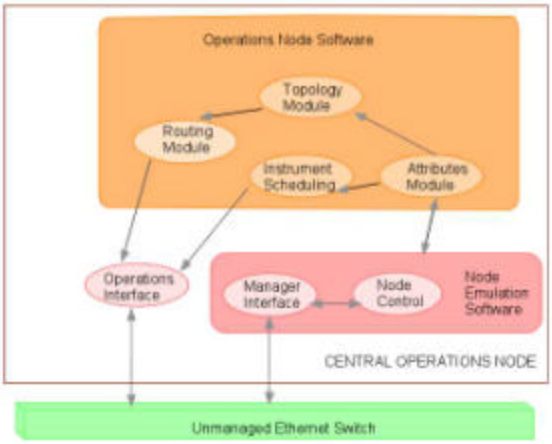

Fig 3. SBI Operations Node Software Modules.

There are three factors upon which the topology algorithm determines connections:

1. Occlusion by Earth i.e. Loss of *Line of Sight (LOS).* The decision for connections can be made on whether the satellites and the ground stations can see each other or not.

2. Link capacity and node switching capacity. Each connection requires a dedicated bandwidth and switching resources. In case of multiple connections on one node, the switching capacity of the node in addition to link capacity has to be considered. The topology module decides any new connection only after considering the total bandwidth available for each satellite.

3. Duration of Line of Sight between two nodes. A longer duration indicates a longer dedicated connection, which reduces route changes.

3.2.3 Emulation Manager

The Emulation Manager controls a defined experiment based on a scenario. Figure 4 shows the components of the Emulation Manager.

The major components of the Emulation Manager software are:

1. The *User Interface* Manager is responsible for defining scenarios and the interactive control of the emulation experiment. During an experiment the *Config Module* retrieves data from the scenario configuration files and stores it in the *Node Database.* The *Log Module* writes out log data periodically to the log files corresponding to the different events in the emulation. The *I/O Module* controls the displays, which are used to interactively manage the experiment.

2. The *Orbital Manager* is responsible for calculating positions of each satellite and ground station in the experiment. We use AGI's *Satellite Tool Kit* (STK) for a satellite database and orbital calculations. From positions we calculate propagation delays for distribution to the SBI

Nodes. This module also houses the *Node Database*, which keeps information such as, node name, type of orbit, mission, positional and vehicular data and data link rates.

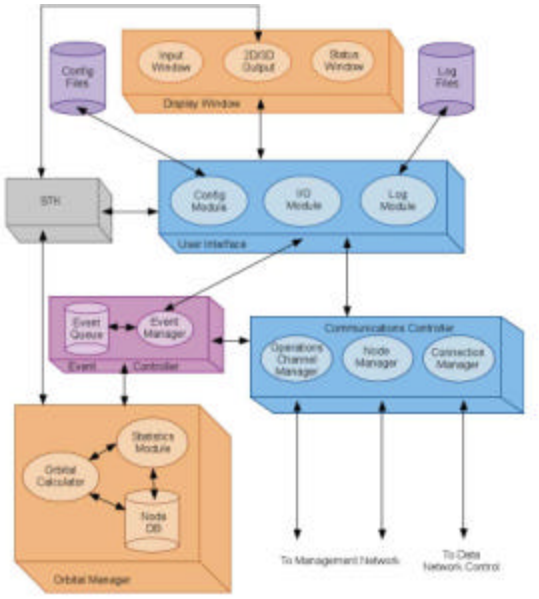

Fig 4. SBI Emulation Manager Architecture.

3. The *Event Controller* maintains an event queue for the experiment. Typical events are instrument start/stop, scheduled link or satellite unavailability, component failure, and so forth.

4. The *Operations Channel Manager* is the medium of communication between the Central Operations Node and the other SBI nodes. The information received from the Central Operations Node consists of instrument scheduling commands and routing information.

5. The *Node Manager* is used to configure the individual SBI Nodes, to monitor their status, and to collect performance information.

6. The *Connection Manager* is used to configure the managed Ethernet switch and change the configuration (i.e. connections) based on the current experiment.

3.3 SBI Emulation System Hardware

The SBI Emulation System is based on an array of PC computers with switched Ethernet interconnection. Our current system consists of:

29 SBI nodes

One (1) Emulation Manager processor

Linux OS on all nodes

Two Ethernet switched networks

6 external 1 Gbps Ethernet ports

6 KVM switches

The SBI nodes form a Data Network based on Ethernet connections through a managed Ethernet switch. We can externally manage the switch to setup/teardown connections. Data communication between the SBI nodes is through this network. A second network is used for emulation management traffic and uses an unmanaged Ethernet switch. This separates the emulation management traffic from the emulated data traffic. Figure 1 illustrates the SBI Emulation System Architecture.

4 CONCLUSIONS

The SBI emulation architecture is a testbed for evaluating SBI network software and satellite communications system performance. The emulation system software emulates those portions of satellite hardware and communications systems that cannot easily be constructed or deployed. The emulation system runs real software code, real data, applications and services to model the satellite system scenario. The emulation system also facilitates multiple connections on a single node and also emulates the communication link to model space communication. Finally, the system is modular so alternative control or management modules can easily be inserted and tested.

5 REFERENCES

[1] Destination: Earth, The Official Website for NASA's Earth Science Enterprise, http://www.earth.nasa.gov/science/index.html

[2] NASA's Earth Observing System (EOS) Home Page, http://eospso.gsfc.nasa.gov

[3] TDRSS Online Information Center, http://nmsp.gsfc.nasa.gov/tdrss

[4] R., Jonkman J. T., *Netspec: Philosophy, Design and Implementation,* MS Thesis, University of Kansas, February 1998.

[5] J. B. Evans, G. J. Minden, G. Prescott, K. S. Shanmugan, V. S. Frost, D. W. Petr, and R. Plumb, "The Rapidly Deployable Radio Network", IEEE Journal of Selected Areas of Communications, Vol. 17, No. 4, pp. 689-703, 1999.